Use of Java Cards in a telematic voting system

Justo Carracedo Gallardo[1], Ana Gómez Oliva[2], Emilia Pérez Belleboni[3], Jesús Moreno Blázquez[4], Sergio Sánchez García[5]

Universidad Politécnica de Madrid, España
EUITT. Telecomunicación
+34 91 336 78 02
{carracedo[1], agomez[2], belleboni[3], jmoreno[4]}@diatel.upm.es
ssanche[5]@proyectos.diatel.upm.es

Abstract

This paper presents a general view of the telematic voting system developed by its authors, with a special emphasis on the important role that smart cards play in this scenario. The use of smart cards as basic pieces for providing secure cryptographic operations in this type of voting scheme is justified. The differences and advantages of Java Cards in comparison with the "classical" smart cards (those that completely conform to the ISO/IEC 7816 standard) are also discussed. As an example, the paper describes one of the applets implemented in the voting Java Card as part of the general telematic voting application.

1. INTRODUCTION

Telematic voting systems are electronic voting systems in which the ballot box is remote and the voter uses computer networks to deliver the vote. This voting system provides the voters with many benefits, such as the ability of issuing the vote from many different voting points and the possibility of getting the election result quickly. Nevertheless, both voters and political parties do not accept this kind of voting system yet. The main reason of this rejection lies on the fact that they do not trust this type of system (Mercuri, 2001) (Internet Voting Report, 2000).

Telematic voting systems, as other conventional voting methods do, must provide the users with some fundamental guarantees of security. It is necessary to assure the authentication of the voters and to simultaneously protect the anonymity of the voter at the moment of issuing the vote. It is also necessary to assure that each voter can only vote one time. Nevertheless, telematic voting systems must not only emulate the conventional voting methods but they must also be able to

handle new threads that may come from a malicious person, due to the fact that programs and digital information can be easily modified. In the design of telematic voting scenarios, this circumstance obliges to include appropriate countermeasures to address these risks and consequently to inspire the confidence of the citizens. For this reason, it is very important for the voters to have some type of *receipt* or digital "supporting document" that allows them to verify the vote that they have cast. In the telematic voting scenario developed by the authors (the VOTESCRIPT system[1]), the smart card is an essential and constituent element of the system. All the participants in the system, both voters and management authorities, need to use their personal smart cards to carry out the tasks defined by the system.

As elements of the general system architecture, smart cards have three essential functions:

a) To guarantee the authentication of the voter. Based in a set of keys and personal data stored in the card, the voter is able to demonstrate his or her right to participate in the election. Similarly, the different management authorities and supervisors of the system have their own smart cards to guarantee the proper authentication.

b) To be a reliable device to carry out certain cryptographic operations. Smart cards that are able to execute public key algorithms strongly guarantee the security of the operations and the privacy of the voters, facilitating the anonymity of the chosen option.

c) To properly protect personal data and keys. The operations of ciphering and signing using the private key are always carried out inside the card. As smart cards are tamper-resistant, keys and other sensible information generated during the voting process can't be illicitly extracted from them.

In this paper (in section 3 and 4) we discuss the restrictions of the use of "classical" smart cards (those which completely conform whit the ISO/IEC 7816 standard (ISO/IEC 7816)), and we present the benefits provided by Java Cards. Java Cards are a valuable tool to improve the security of the system. Indeed, these cards offer system developers the facility of storing small applications (applets) in the memory card with the guarantee of integrity and inviolability that every confidential data stored in a smart card has. To emphasize the important role played by smart cards in the global behaviour of telematic voting systems, in section 2 of this paper we present a summarized description of our voting system, the VOTESCRIPT system. Finally, in section 5, we describe the implementation of one of the operations carried out by an applet in the voter smart card. This

1 This system has been developed within VOTESCRIPT projects (Secured Electronic Voting based on Advanced Cryptography) sponsored by National Council for Science and Technology of Spain (TIC2000-1630, TIC2002-4223 and TIC2003-2141)

description illustrates the structure and functionality of a APDUs (Application Protocol Data Units) used in communications between the applets stored in the smart card and the computer that acts as a voting terminal.

2. SCENARIO OF COMMUNICATION

The VOTESCRIPT system (Carracedo et. al, 2003) tries to support telematic voting in environments where all the votes are collected in a single ballot box, although we are working on its adaptation to environments in which the presence of multiple ballot boxes would be necessary.

2.1 AUTOMATIC AGENTS AND SYSTEMS

In the VOTESCRIPT communication scenario (Figure 1) a set of automatic systems takes part:

- Authentication Points (APs). They are kiosks equipped with a card reader, wherein the voter is authenticated by the system as the first step of the voting process.

- Ballot Points (BPs). They are cabins equipped with a card reader that helps the voter to issue his vote. In VOTESCRIPT, the voter can choose, during the authenticating process, any of the multiple existing APs and is able to issue the vote in any of the existing BPs.

- An Administrator of authentication. This automatic agent sends back to the voter an authorization ticket that enables him to issue his vote on the Ballot Point.

- Several Intervention Systems that complement the work of the Administrator. Every Intervention System is controlled by an Inspector, designated by each of the groups of voters or candidatures authorized to supervise the voting process.

- A Ballot box that gathers the votes and gives back voting receipts to the voters.

- A Counter for tallying the votes once the election has finished.

- Verification Points which allow the voter to verify that his vote has been considered and counted correctly.

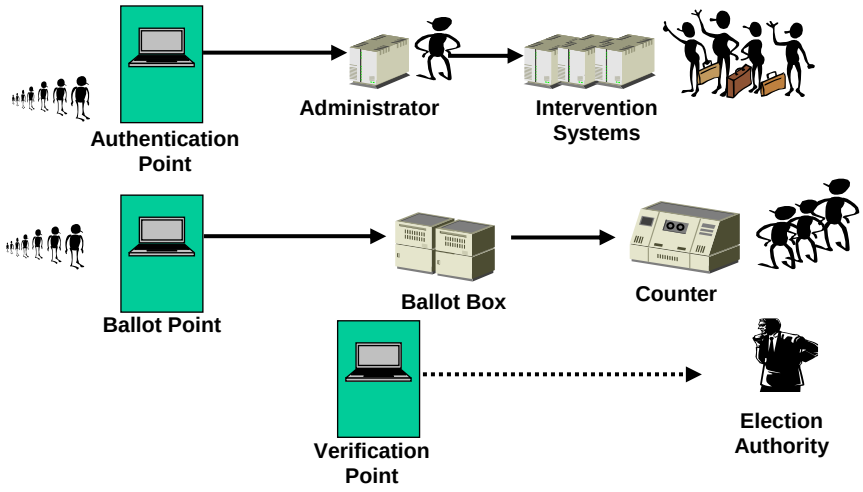

Figure 1. VOTESCRIPT system architecture

2.2 PARTICIPANTS

The people who participate in the process are:

- Voters. Every voter is in possession of a voting smart card that has been specially defined for VOTESCRIPT system. This card is able to carry out several cryptographic operations.
- A Manager of the Administrator System.
- Inspectors. Each Inspector manages one of the Intervention Systems.
- An Election Authority who is in charge of the general control of the system and responsible to solve possible claims.

The Administrator and the Intervention Systems, along with the Manager of the Administrator System and the Inspectors, constitute the equivalent, in this telematic system, to the conventional electoral board.

2.3 KEYS AND IDENTIFIERS

In a previous stage, before the election begins, each voter receives one voting smart card and a voter's identifier. The voter must go to a specially assigned sites where the APs and BPs are installed, using his smart card to get the proper authorization from the Administrator and to thus be able to cast his vote.

The Administrator, the Intervention Systems, the Ballot box, the Counter and the Election Authority have a pair of keys (public and private). All these public keys are known by all telematic agents and all participants in the voting system. In addition, the voters have a pair of keys (public and private) stored in their cards before the election begins. The certificated public key is well known by all the telematic agents of the system and by the Election Authority.

2.4 GLOBAL BEHAVIOUR OF THE SYSTEM

Due to space constraints, we will not describe in this paper the detailed secured protocols through which the different system's agents communicate. However, in order to understand the kind of tasks that have to be carried out using smart cards, a brief description of the global behaviour of the voting system that has been developed is presented next.

First, the voter inserts his smart card into the Authentication Point in order to get the authorization to issue his vote. In order to do that an application, which could be called *voter's application*, has been developed. This application will interact with both the voter and the Administrator agent. This agent is in charge of verifying the voter's identity and checking that he has not yet voted. All the Intervention Systems carry on the same tasks although there is not direct contact between them and the voter. In case of positive matching, the Administrator will send back to the voter the requested authorization which will be stored in the smart card.

Next, the voter will head for the Ballot Point carrying his card, where the *voter's application* will ask him for the voting option. Afterwards, the selected option together with the authorization received in the previous step, will be sent to the Ballot Box.

In order to allow the voter to verify that his vote has been properly considered, the Ballot Box will send him back a receipt with the information regarding the issued vote. At the end of the process, this receipt –stored in the card- provides the voter with enough proof to make a claim in case of disagreement with the results.

Once the voting period has finished, a direct intervention of the Inspectors, the Manager of the Administrator System and the Election Authority will start the process of the Counter agent, guaranteeing the tallying process and the publication of the final results.

Afterwards, a short period of time for the verification procedures will be opened, that is when the receipt could be used in case of disagreement with the results.

3. VOTING WITH SMART CARDS

3.1 THE NEED TO HAVE AN ELECTRONIC SUPPORT FOR SECURED STORING: USE OF CLASSICAL *SMART CARDS*

As mentioned in the preceding section, every entity participating in the ballot, i.e. voters and voting systems agents, must have its corresponding pair of a secret key and a public key (the last one provided in a certificate format). To guarantee the security of the voting system, voters and participant agents must keep the keys safe in an adequate manner.

Moreover, as indicated previously, during the voting process two pieces of information are created (authorization ticket and voting receipt), both of them protected by cryptographic algorithms. These pieces must be stored in some kind of reliable and easily transportable support that could assure the availability of this information when required in the voting process.

For that reason, it is necessary to be ready to have an electronic support for a secure and strong storage: we have concluded that a smart card is the most appropriate device to support this kind of storage. The smart card allows, on one hand, to keep safely the private keys of voters and agents so as to guarantee their identity; on the other hand, it permits to store the pieces of information that are generated along the process.

In the initial stages of the development of the project the "classical" smart cards were used (those fully in accordance with the ISO/IEC 7816 standard). These cards were employed to perform two main actions: a) to hold the identifier and keys of the voters, as well as the information related to the voting process; b) to perform the signature/cipher of the information using the microprocessor that is included in this type of card. In this way it was possible to have a cheap and portable device which prevents the information stored inside from being read or modified in a fraudulent way.

As mentioned before, the *voter's application* is in charge of dialoguing with the voter to ask for his voting option and to carry out the secured dialogue among the different system's agents. This process is undertaken by means of appropriate cryptographic operations.

Due to the restrictions of the classical smart cards, in the initial stages of the development of the project, the voter's application was located within the Authentication Point and the Ballot Point. This means that some cryptographic operations needed to authenticate the voters and to handle their votes must be carried out in these Points. However, there is a risk that malicious software loaded on the Authentication Point or the Ballot Point can change the voter's vote without the voter noticing it (Rubin, 2001). Therefore, the loss of security during the whole process could be a real risk.

3.2 IMPROVING THE SECURITY: JAVA CARD

The way to improve remarkably the security of the system is to make the most part of the *voter's application* reside within the smart card. From a practical point of view, this solution is not feasible using a conventional smart card, but it can be obtained using a new generation of smart cards: Java Cards. In fact, these cards, besides pooling all the security requirements of the conventional smart cards, also allow the storage and later execution of different user applications, developed in Java language. This means that it is the suitable place for the *voter's application* due to the new features of Java Card combined with the tamper-proof characteristics of any type of smart card. Therefore, this solution provides users with all the benefits of Java Card plus a high level of security and reliability.

The introduction of Java cards allows improving the user's authentication process. An inherent previous task in the use of smart card lies in the guarantee that the person who presents the card is, in fact, his rightful owner. In the previous version of the voting system, using classic smart cards, this authentication was obtained by means of a PIN. Using Java Cards, this mechanism is substituted by a stronger procedure, based on the existence of an applet for the storage and latter verification of the participants' fingerprints (Maltoni et al., 2003). The biometric applet used in this system is Precise Biomatch™ J from Precise Biometrics. This applet is able not only to store the fingerprint of the card's owner but also to implement the algorithms to verify (within the card) its validity (match-on-card).

4. JAVA CARD TECHNOLOGY

In order to facilitate the subsequent explanation a brief introduction to Java Card technology is included in this section.

A Java Card, like any other smart card, is a chip card with a microprocessor inside. The integrated circuit incorporated in the card contains a CPU and elements used for data transmission and storage. Essentially a Java Card is a smart card which is able to run applications, called applets, written in Java programming language. These applications run in the Java Virtual Machine called JCVM (Java Card Virtual Machine) (Hansmann et al., 2002) (Carracedo, 2004) (Ferrari et al., 1998).

The physical appearance of the plastic substrate is fully compatible with those conventional smart cards which are in compliance with the international standard ISO/IEC 7816. Also the microprocessor architecture and the organization and the functionalities of the external contact points conform to that standard.

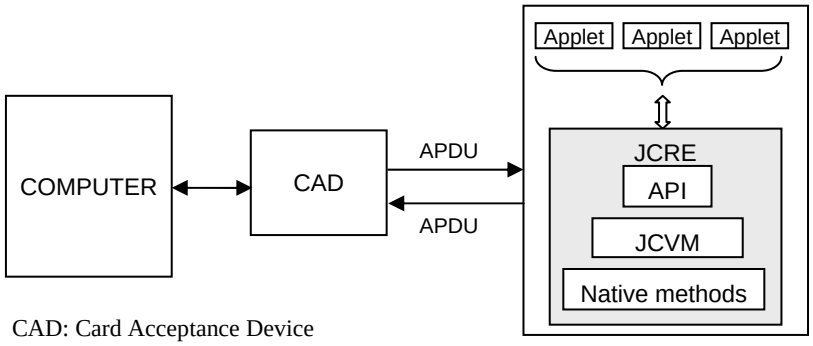

Figure. 2. Java Card logical architecture

Figure 2 shows a simplified view of the Java Card logical architecture. The applications and data will be stored in the memory and the machine code will be executed by the microprocessor. Each manufacturer will incorporate a proprietary microprocessor with a proprietary internal architecture; therefore every program will run using a specific set of machine instructions. For that reason, the "native methods" box that appears in Figure 2 will match the particular features of each microprocessor and will support the JCVM as well as other *classes* of the Java runtime environment.

JCVM interprets *bytecode* instructions, executes applets, enforces runtime security and manages objects and memory.

The API (Application Programming Interface), logically located on the next upper layer, contains those *classes* which provide the supported services and the system applets themselves. The API includes the functions that could be invoked by the user applets as well as the required applications to download and install these applets in the card.

The three layers (native methods, JCVM and API) constitute the card execution environment or JCRE (Java Card Runtime Environment) that is in charge of the program execution and resource management. In other words, it is a sort of card operating system.

To adapt a Java card to a specific application, the applets are firstly developed, implemented, tested and debugged in a conventional PC. After that, several applets can be installed in the card. As well as the standard smart cards, Java cards also hold a small room for memory and so, they only support a subset of the features of Java language.

Concerning the APDU (Application Protocol Data Unit) set defined in ISO/IEC 7816-4, Java cards exclude APDUs related with file system access (internally controlled by the JCRE), and add new ones to support communication between applets and external programs.

4.1 APDUs

As mentioned before, communication with Java cards is carried out by using APDUs. There are two different types of APDUs: *commands*, received by the card; and *responses*, sent by the card as a result of a command execution. The format of the two types is shown in Figure 3.

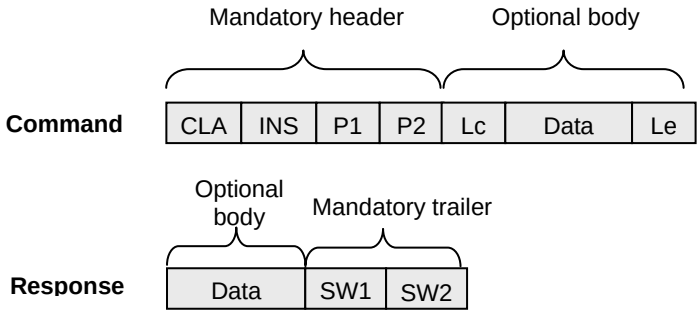

Figure 3. APDUs structure

4.1.1 COMMANDS

The commands have two parts: header and body (see Figure 3). The header is mandatory in every command. It consists of 4 octets with the following meaning:

CLA: *Class of instruction*. It identifies a type of command.

INS: *Instruction code*. It identifies the specific instruction of the command.

P1 and *P2*: *Parameters 1* and *2* provide further qualification to the instruction.

The body has variable length and it is optional. *Lc* (1 octet) specifies the length of the *Data* field in octets. *Data* contains the information sent to the card to execute the instruction given in the header. *Le* (1 octet) indicates the number of octets expected in the response *Data* field returned by the card.

4.1.2 RESPONSES

As in commands, the responses have two parts: one mandatory and the other optional. The mandatory part consists of 2 octets, *SW1* and *SW2*, called *status words*, which represent the execution result, that is to say, the card status after the command execution. For example, the values *SW1*=0x90 and *SW2*=0x00 indicate that the command execution has been successfully completed in the card. The *Data* field is the optional part and it contains the results of the requested operation. The length of this field is constrained by the *Le* field in the command that originated this response.

There are four possible combinations of commands and responses in APDUs, depending on whether the *Data* field is present or not; as it is shown in Figure 4. The use of a given combination will be determined by the communication needs.

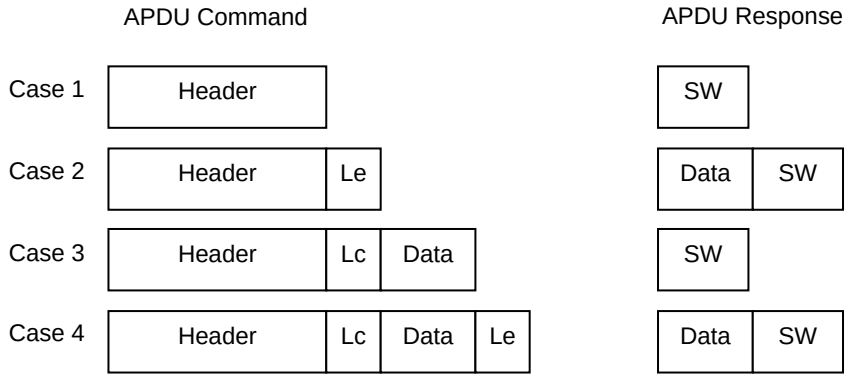

Figure 4. Combinations of commands and responses

5. EXAMPLE OF AN OPERATION CARRIED OUT BY THE JAVA CARD

The following description details an example of the operations executed by the Java Card. This example is presented as a case study to illustrate the power of the operations that can be carried out using Java Card technology. Particularly, we describe the operations between the smart card and the Authentication Point in order to allow the voter to get the authorization to vote from the Administrator.

All the operations that make possible the interaction among the voter, his or her Java Card and the Administrator in charge of the voters authentication are described. Figure 5 shows those interactions across the Authentication Point.

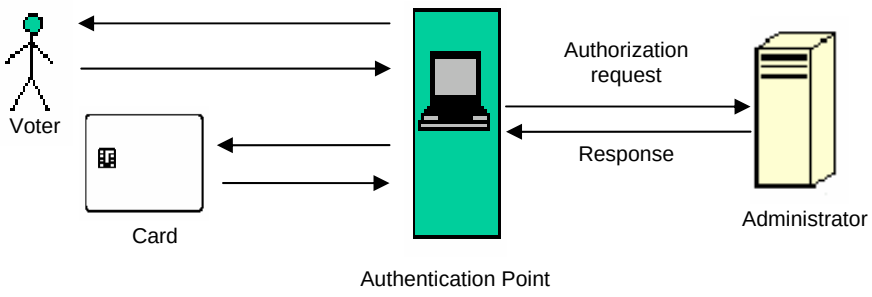

Figure 5. Simplified scheme of the interaction among the voter, his Java Card and the Administrator

On behalf of the voter, his smart card will fulfill all the tasks needed to generate the pieces of information that are going to be sent to the Administrator, so that all

critical information is processed inside the card in a protected mode that ensures the confidentiality of both data and operations.

The dialog between the Authentication Point and the Administrator is performed by means of messages, which are the result of the application of complex cryptographic operations. The Authorization request is a piece of information containing the following information:

$$Ad_P [k_{dV}, V_S (k_{dV}), Id.Voter, V_S (Id.\ Voter)] \quad (1)$$

$Ad_P [\ ]$ It represents that all the information between the square brackets is ciphered with the Administrator's public key. This information is made by the concatenation of all the elements separated by comas.

K_{dV} It is one of the keys generated inside the Java Card processed by cryptographic algorithms so as to prevent from further recognition of it that would link voter and vote.

$V_S (\ )$ It represents that the information between the brackets is ciphered with the Voter private key

Id. Voter It is the identification of the voter.

To generate this piece of information and to obtain it from the card an applet must be installed on the Java Card in order to send the proper message to the Administrator. The applet must allow the execution of all the steps needed neither risking for data nor operations.

5.1 DESCRIPTION OF THE APPLET IMPLEMENTED ON THE JAVA CARD

The authorization request is a complex piece of information that is generated inside the card using its cryptographic features. As it has been previously stated, this piece of information has to be sent to the Administrator through the Authentication Point, but the Java Card has a narrow capacity of input/output buffering. Consequently, after the information is generated, a number of interactions will be required in order to extract the whole information from the Java Card toward the Authentication Point.

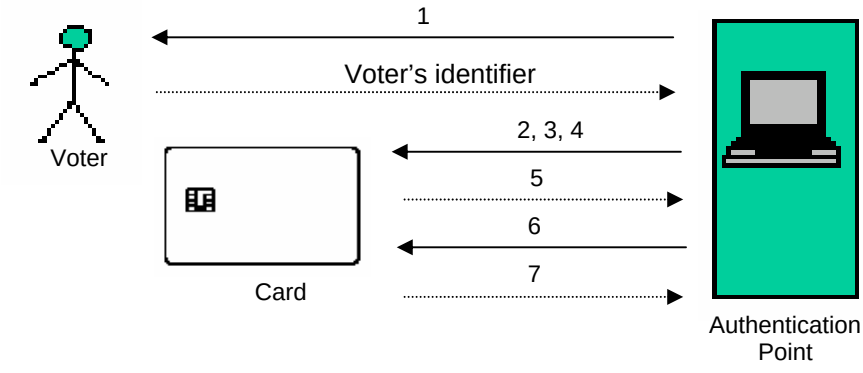

Figure 6. Sequence of steps needed to generate the Authorization Request operation

Figure 6 shows the complete process to obtain and send the data to the Administrator.

1. *Authentication Point → Voter:* Ask for the voter's identifier.
2. *Authentication Point → Card:* Ask for the generation of the keys to be used to cipher and decipher the vote.
3. *Authentication Point → Card:* Ask for the storage of the provided Administrator's public key.
4. *Authentication Point → Card:* Ask for the generation of the *Authorization Request*.
5. *Card → Authentication Point:* Answer to the request in the step four. If the operation was successfully executed, the piece of information consists of a number of blocks of 128 octets. The total number of blocks is reported in one of the parameters of the answer. If the operation could not be executed, such event is reported in a separated parameter.
6. *Authentication Point → Card:* i^{th} block request.
7. *Card → Authentication Point*: i^{th} block is sent back or an error is reported.

   Steps 6 and 7 are repeated until the whole message has been transferred.

This sequence of operations shows the case in which a specific operation is carried out across the Authentication Point. The applet installed on the Java Card is able to return a number of octets in response to the proper request previously made by the Authentication Point. More than one interaction (step 6 and 7) will be needed due to the fact that the regular size of the Java Card input/output buffer is smaller than the regular size of the message –in spite of being 255 octets the maximum available in the chosen card, in this development we make use of only 128 octets length so that performances are improved. The length of the message depends on the size of the ciphering keys. The RSA keys in this development are of 1024 bits, which result in a message of around 600 octets. The fact that the generation and extraction of data are separate operations forces the applet to require permanent storage space inside the card. Otherwise, the confidentiality of data would not be ensured. One of the most serious limitations of current smart card technology is the short memory space they have to store data. We have used the Sm@rt Café Expert Java Card from Giesecke & Devrient that implements a garbage collector in order to prevent the memory space limitation from being even more critical. Proceeding with large blocks of data that requires intermediate storage space for the operations could be an important drawback depending on the card that is used.

To summarize: in order to generate the appropriate piece of information inside the Java Card and to send it to the Administrator, two main operations are required:

a) Generation of the information piece.

b) Extraction block by block of the whole generated piece of information.

The information that must be stored in the card is briefly commented in order to proceed with a detailed description of those operations.

5.2 DATA STORED ON THE JAVA CARD

Before generating the piece of information inside the Java Card, the voter identification and some keys must be supplied. The voter identification is the value of one parameter given to the applet by the APDU command related to the Authorization Request (step 4, Figure 6). Some of the keys needed for different operations are internally generated when required; whereas other keys are externally supplied as is the case of the Administrator public key in the previous example.

For security reasons, the voter's private and public keys must be generated inside the smart card to ensure a complete confidentiality of the private key that strongly identifies the identity of the voter. Private keys will never be read from the card, being only the applet able to manage them. This pair of keys must be generated in a stage previous to the electoral process.

Asymmetrical keys used for ciphering and deciphering the vote are also internally generated because of similar considerations to those stated for voter's private and public keys.

There are two possible ways to provide the external public keys and the voter identifier. It is possible either to load it together with the applet, or to send it as a value of a parameter when the operation is requested. Our development makes use of the second option although we are currently working on the first one.

5.3 DESCRIPTION OF THE GENERATION OF THE AUTHORIZATION REQUEST

After receiving the APDU described in the previous paragraph, the applet begins the execution of the respective method. Inside the applet, there is a general method, called *process*, which is in charge of determining the type of operation to execute (from parameter INS of the APDU) and therefore the method that must be invoked.

```
 /**
 * Method that processes an incoming APDU.
 * @see APDU
 * @param apdu incoming APDU
 * @exception ISOException according to ISO 7816-4
 */
public void process(APDU apdu) throws ISOException {
    // gets the APDU
    byte[] apduBuffer = apdu.getBuffer();

    // select the method according to INS parameter
    switch (apduBuffer[ISO7816.OFFSET_INS]) {
              .
              .
              .
       case INS_BUILT_ADMIN_DATA:
         // checks if command is correct for this applet
            if (apduBuffer[ISO7816.OFFSET_CLA] != VOTER_CLA)
                ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
```

```
                builtAdminData(apdu);
                break;
                  .
                  .
                  .
            default :
                // Not supported command
                ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
                break;
        }
}
```

The following operations are executed by `builtAdminData` in order to generate the piece of information addressed to the Administrator (this expression is shown in expression (1), Section 5):

a) Verification of presence of the required keys.

b) Initialization of the cipher to operate with the voter private key.

c) Ciphering of the voter identifier received in the *request*.

d) Ciphering of the K_{dV} key.

e) Concatenation of the voter identifier, the ciphered voter identifier, the K_{dV} key and the ciphered K_{dV} key.

f) Initialization of the cipher to operate with the public key of the Administrator.

g) Ciphering of the result of the concatenation

5.4 OPERATION: SENDING OF THE GENERATED AUTHORIZATION REQUEST

This operation permits to obtain all the blocks –one by one– belonging to a piece of information generated inside the card. Since the result must be divided in 128 bytes blocks, the P1 parameter of the APDU indicates which block must be sent back.

5.5 SEQUENCE OF APDUS

Sm@rtCafé Professional Toolkit 2.0 from Giesecke & Devrient manufacturer has been used for the applets development. This tool provides the necessary elements for an applet development, such as card simulation, debugging, loading and verification of operation in the card. In Figure 7, obtained from the toolkit, the sequence of APDUs is shown. Numbers from 2 to 7 have been overlapped in order to show the correspondence between the steps numbers used in Figure 6 (step 1 is omitted because it does not involve using of Java Card).

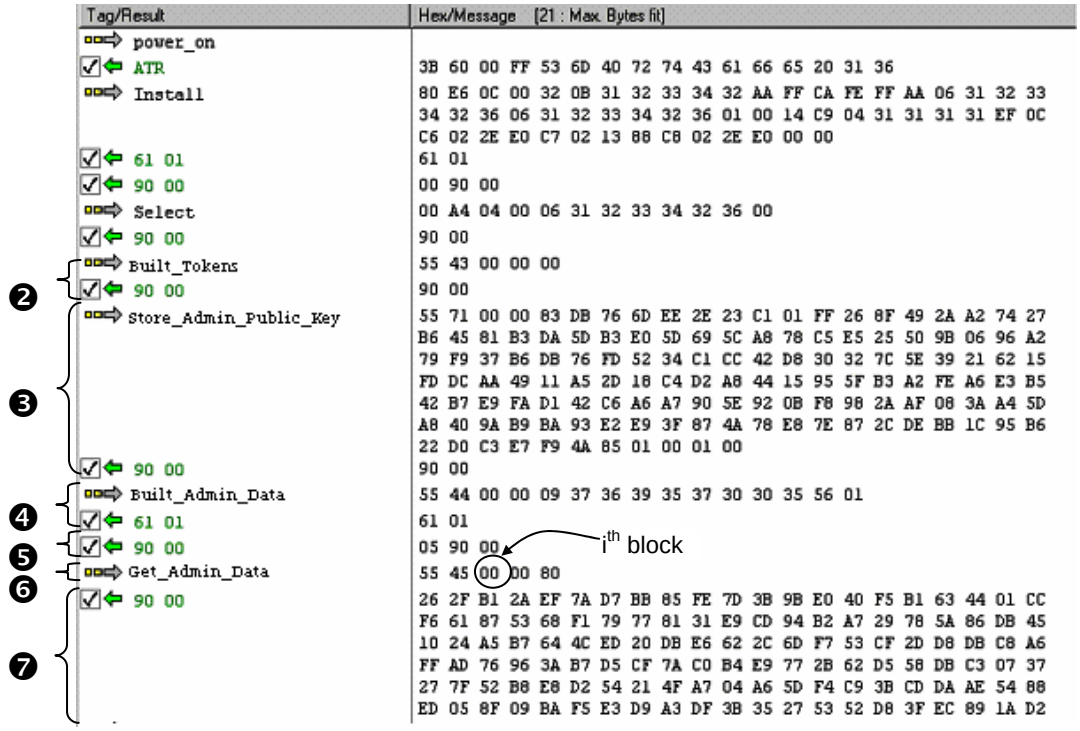

Figure 7. Sequence of APDUs

Analyzing in detail one of the APDUs shown in Figure 7, i.e. the command that requests the preparation of data for the Administrator (operation 4), the different parts described in section 4.1 can be identified (Figure 8).

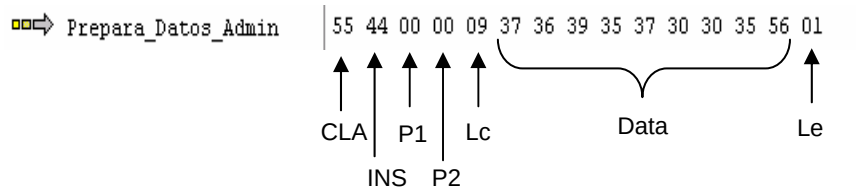

Figure 8. Example of the format of the APDU corresponding to the command Request of preparation of data for the Administrator

As it can be noticed, the instruction is identified by means of a CLA=0x55 and a INS=0x44, so that the applet can determine which is the operation to execute. Parameters P1 and P2 that take the value 0x00, provide no further information. Because of the data contained by the APDU, we use the *Lc* field whose value (0x09) indicates that the field of data is composed of nine bytes. These nine bytes following the *Lc* parameter contain the voter identifier used by the applet to prepare the information to be sent to the Administrator. The last byte -the *Le* field- indicates that only one byte of data is expected in the answer to this command.

Operation 5 shows the answer announcing that requested data will be completed with five blocks.

Operation 6 shows the request of the first block (00) of the information piece to be sent to the Administrator, and operation 7 represents the response with the respective block. Next, the Authentication Point by means of new operations (similar to 6), will ask for subsequent blocks (00, 01 ... 04) and the card will answer by sending the proper response like 7.

Through this example, we can observe the nature of atomic operations that are needed to implement the dialogue between the smart card and the computer which acts as a terminal of the system.

As we stated in Section 3, Java Cards easily permit *voter's applications* to be implemented by means of applets and they also permit to store this applets within memory card. The examples described above illustrate the behaviour of an applet that supports parts of a *voter's applications* stored in a Java Card and demonstrates the feasibility of this kind of solution.

6. CONCLUSIONS

In telematic voting applications, the use of smart cards operating under public key algorithms offers great advantages to guarantee both the voting anonymity and the voter's authentication. Since they are tamper-resistant, smart cards effectively protect personal keys of voters and the *receipts* generated after the voting.

Smart cards that we call *classic* (those which conform to the ISO/IEC 7816 standard) allow to easily introduce data in the customization phase but offer few opportunities for application designers to load new programs into the card memory.

A new generation of smart cards, Java Cards, do allow to introduce in the card memory small applications (*applets*), codified in Java, which support most of the needed cryptographic operations, maintaining in total secrecy the keys used for such operations.

Although the small size of Java Cards' memory imposes certain limitations regarding the operations that can be carried out, adequate design and proper usage of existent tools permits to carry out complex and robust operations, which guarantee the global security of the system.

7. REFERENCES

Carracedo J., Gómez A. and Carracedo J.D., (2003). Sistema VOTESCRIPT: Una propuesta innovadora desarrollada para resolver los problemas clásicos de votación electrónica. 2º Congreso Iberoamericano de Seguridad Informática (CIBSI'03). México D.F., (in Spanish).

Carracedo J., (2004). Seguridad en redes telemáticas. Mcgraw-Hill. ISBN 8448141571 (in Spanish)

Ferrari J. et al., (1998). Smart Cards: A Case Study. IBM International Technical Support Organization. http://www.redbooks.ibm.com

Hansmann, U. et al., (2002). Smart card application development using JAVA. Springer-Verlag. ISBN 3540432021.

Internet Voting Report, (2000). California Internet Voting Task Force. A Report on the Feasibility of Internet Voting. http://www.ss.ca.gov/executive/ivote/final_report.htm

ISO/IEC 7816 Identification cards – Integrated circuit(s) cards with contacts
ISO/IEC 7816-1 (1998). Part 1: Physical characteristics.
ISO/IEC 7816-2 (1999). Part 2: Dimensions and location of the contacts.
ISO/IEC 7816-3 (1997). Part 3: Electronic signal and transmission protocols.
ISO/IEC 7816-4 (1995). Part 4: Interindustry commands for interchange.
ISO/IEC 7816-5 (1994). Part 5: Numbering system and registration procedure for application identifiers
ISO/IEC 7816-6 (2004). Part 6: Interindustry data elements for interchange.
ISO/IEC 7816-7 (1999). Part 7: Interindustry commands for Structured Card Query Language (SCQL).

Maltoni D. et al., (2003). Handbook of Fingerprint Recognition. Springer-Verlag. ISBN 0387954317.

Mercuri R., (2001). Testimony presented Mercuri R. to the U.S. House of Representatives Committee on Science. http://www.house.gov/science/full/may22/mercuri.htm

Rubin A. (2001), AT&T. Labs-Research. Florham Park, NJ. http://avirubin.com/e-voting.security.pdf